A STUDY ON LOG EVENT NOISE REDUCTION BY USING NAÏVE BAYES SUPERVISED MACHINE LEARNING

By

MARC SPAIN

Bachelor of Science in Microbiology

Bachelor of Science in Botany

University of Oklahoma

Norman, OK

1982

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2019

A STUDY ON LOG EVENT NOISE REDUCTION BY USING NAÏVE BAYES SUPERVISED MACHINE LEARNING

Thesis Approved:

Dr. Nohpill Park

Thesis Adviser

Dr. Chris Crick

Dr. Esra Akbas

ACKNOWLEDGEMENTS

I would like to thank my wife who has stood by me as I have traveled down this journey.

I would also like to thank Dr. Park who listened to my ideas as I worked out the research that I wanted to do for this thesis. He has encouraged me at every step and kept me going during the times I was ready to give up.

Name: MARC SPAIN

Date of Degree: DECEMBER, 2019

Title of Study: A STUDY ON LOG EVENT NOISE REDUCTION BY USING NAÏVE BAYES SUPERVISED MACHINE LEARNING

Major Field: COMPUTER SCIENCE

Abstract: This research addresses which Naïve Bayes model would be best to predict Windows log events that could be considered noise or in other words not containing information about malicious activities. With the exploding amount of log data being generated by servers, large corporations or organizations are having an increasingly difficult time analyzing these logs to find evidence of malicious activity in their environment. Fortune 200 and larger corporations today are producing Terabytes of log events daily and this is expanding at a rate that soon it will be in the Petabytes. It is estimated that 80 to 90 percent of these log events could be classified as noise or just informational. They are not needed for finding evidence of malicious activity. By showing a process that can be used to predict whether these log events are noise or non-noise, with a reasonable degree of accuracy, tools could then be used to analyze log events to find malicious activity to filter out noise events and reduce the amount of data needed to be processed. This research will compare the Naïve Bayes Bag of Words Multinomial, Multinomial TF-IDF and Multi-Variate Bernoulli models using different size feature word sets in predicting Windows noise log events.

.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

The enterprise organizations of today are powered by data. They take information in, analyze it, manipulate it, and create more as output. One of the key areas of the enterprise organization is Information Technology (IT) security. One key part of IT security are log events generated by servers and other security appliances. Every log event has something to tell but some are more important than others. A log event needs to be analyzed in order to know or determine its level of importance. The faster this can be done and the earlier in the collection process, the more agile and responsive IT security can be. The more efficient log classification is less effort is needed to be spent on analyzing logs for malicious activity. Resources can then be focused toward alerting and response. How log events are classified becomes nearly as important as the logs themselves. In today's world, log event collection from IT infrastructure has become a security requirement and is a complex resource intensive process. The number of log events being collected has become very large and is growing day by day. Security departments in these enterprise organizations use specialized software to data mine and run analyses on these events. The purpose of which is to discover security problems or malicious activities. Most of these log events are just informational and not useful in finding malicious activities and are considered noise. As the number of events collected increases, the time and resources needed for these analysis efforts grow. I propose it has become important to determine which events are not useful and can be considered noise. This research will show that by using Naïve Bayes supervised

machine learning algorithms, noise log events can be classified and tagged and thereby reduce the amount of log events needed to be analyzed for security needs. For the purpose of this research I intend to limit the events to be tested to only Windows servers and their logs.

1.1 Security Logs and Security Log Management

Security event logs consist of information that is used to track associated user and system activity. In conjunction with appropriate tools and procedures, logging can assist in detecting security violations, troubleshoot system performance problems and system vulnerabilities. Log management ensures that computer security records are stored in sufficient detail for an appropriate period to allow log reviews when needed. Log reviews identify security incidents, policy violations and fraudulent activities which support audit activities and internal security investigations. Log event collection is required in order to comply with federal, state, local laws and regulations as well as data security guidelines. Retaining logs allows for the fact that it often takes a while to notice that a compromise has occurred or is occurring and allows security teams a sufficient log history to better determine the length of time a potential unauthorized activity has been occurring and the potential impact to an enterprise organization system.

1.2 Data Security Standards and Guidelines

Payment Card Industry Data Security Standard (PCI DSS): PCI DSS applies to organizations that store, process or transmit cardholder data for credit cards. One of the requirements of PCI DSS is to track access to network resources and cardholder data.

Federal Information Security Management Act of 2002 (FISMA): FISMA emphasizes the need for each Federal agency to develop, document, and implement an organization-wide program to provide information security for the information systems that support its operations and assets. It describes several controls related to log management, including the generation, review, protection, and retention of event logs, as well as the actions to be taken in event of log failure.

General Data Protection Regulation (GDPR): GDPR is a regulation in EU law on data protection and privacy for all individual citizens of the European Union and the European Economic Area. It also addresses the transfer of personal data outside the EU and EEA's. The GDPR aims primarily to give control to individuals over their personal data and to simplify the regulatory environment for international business by unifying the regulation within the EU.

23 NYCRR 500 (NYDFS): 23 NYCRR 500 applies to supervised financial entities. It requires entities to assess their cybersecurity risk profiles and implement a comprehensive plan to recognize and mitigate the identified risk.

Gramm-Leach-Bliley Act (GLBA): GLBA requires financial institutions to protect their customers' information against security threats. It specifies that log management can be helpful in identifying possible security violations and resolving them effectively.

Sarbanes-Oxley Act (SOX) of 2002: SOX applies primarily to financial and accounting practices. It also encompasses the information technology functions that support these practices. SOX can be supported by reviewing logs regularly to look for signs of security violations, including exploitation, as well as retaining logs and records of log reviews for future review by auditors.

1.3 SIEM: Security Information and Event Management

SIEM: An approach to security management that combines SIM (security information management) and SEM (security event management) functions into one security management system. The acronym SIEM is pronounced "sim" with a silent e.

The following are a list of core functions found in SIEM tools.

1. Log management: Focus on simple collection and storage of log messages and audit trails

2. Security information management (SIM): Long-term storage as well as analysis and reporting of log data.
3. Security event manager (SEM): Real-time monitoring, correlation of events, notifications and console views.
4. Security information and event management (SIEM): Combines SIM and SEM and provides real-time analysis of security alerts generated by network hardware and applications.

The tools used in SIEM software are a type of centralized logging software. SIEM software collects log data generated throughout the organization's technology infrastructure, from host systems and applications to network and security devices such as firewalls and antivirus managers. The software then identifies and categorizes incidents and events, as well as analyzes them. These analytics can be used to monitor for suspicious or malicious activity and automatically alert on this behavior, providing true real-time and continuous monitoring of logs. The ability to correlate alerts from distinct log sources provides context-rich insights into what is going on in the enterprise organizations information technology environment. Log collection is central for SIEM's to work and be useful to the corporate enterprise. The more log sources that send logs to the SIEM, the more can be accomplished with the SIEM. Corporate networks and servers generate vast amounts of log data. A Fortune 500 enterprise's infrastructure can generate over 10 TB of plain-text log data per month. The company I work for generates 3.7 TBs of plain-text log data a day. The following are SEIM software tools used in this research.

Splunk Enterprise

Splunk Enterprise software platform processes, indexes and stores most forms of data in its native format. It includes data indexing tools, which enable the location of specific data across large data sets.

Splunk Enterprise Security

Splunk Enterprise Security is software that adds additional functionality to the basic Splunk Enterprise software through the use of prebuilt dashboards, alerts and reports. It provides insight

to quickly detect and respond to internal and external attacks and simplify threat management minimizing risk.

Splunk User Behavioral Analytics (UBA)

Splunk UBA is software that mines data from the basic Splunk Enterprise software product and uses machine learning algorithms to find hidden threats and anomalous behavior across users, devices, and applications. Its results provide ratings and supporting evidence. It detects insider threats using extensible unsupervised machine learning algorithms.

The SEIM tools at the enterprise organization that produced the data used in this research require the following resources. These numbers are based on ingestion of 3.7 TBs/day of log events.

	Number of Servers	Total Cores	Total Memory
Splunk Enterprise	83	996	996GB
Splunk Enterprise Security	4	64	128GB
Splunk User Behavior Analytics (UBA)	10	160	640GB

Table 1 Resources Used by SIEM Software Tools

Log Events which are produced by malicious activities are very individualized and unique to the methods used to compromise today's servers. This presents a challenge by security teams engaged in finding and tracking specific log events which would show that a server is under attack by a hacker. Methods to find these malicious events are very complicated and can use large amounts of computing resources. One way to simplify this process would be to determine which log events are not useful or considered noise and then tag them or strip them out of the log stream. Thus any process looking for malicious activities would not have to search or inspect every log event.

1.4 Proposed Research

This research will show that applying Naïve Bayes prediction would be a good process to use in determining noise events produced by Windows event logs. By determining the which Naïve Bayes model has the highest accuracy using the least number of features could be added to a SIEM process to improve the efficiency of that tool. The reduced amount of data processed by the SIEM software tools would reduce their need for CPU and Memory resources needed to find log events indicating malicious activity. Thus presenting a resource and potential cost saving to any company or organization that uses it.

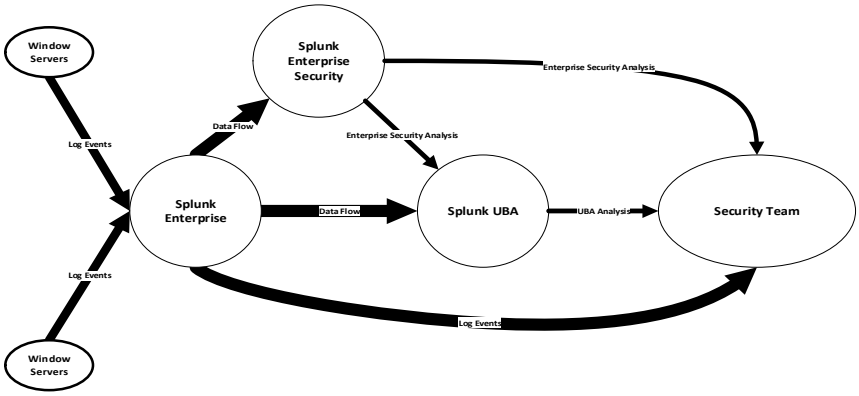


Figure 1 Work Flow of Events Logs through SIEM Tools before Naïve Bayes Log Event Classifier Tagging

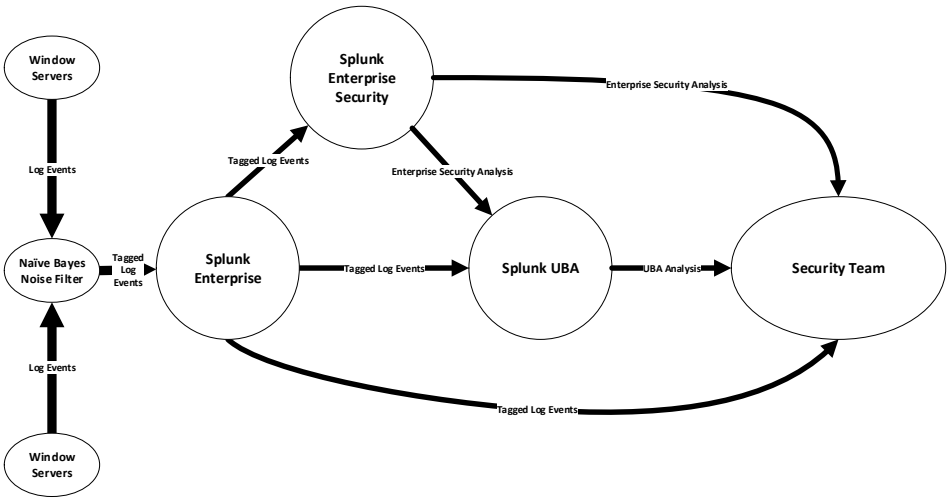


Figure 2 Work Flow of Event Logs through SIEM Tools after Naïve Bayes Log Event Classifier Tagging

Figure 1 shows the work flow of the event logs through the SIEM tools without the noise events being classified and a tag attached to the log. Figure 2 shows the work flow after the noise events have been tagged and the SIEM tools modified to filter out log events without the tags. The width of the lines represents the amount of log events entering and leaving each process of the work flow.

The rest of this thesis is organized as follows: Chapter II introduces the literature reviews performed prior to the start of the research. Chapter III provides a description of the methodology used in the analysis of the Naïve Bayes classification and predictive methods to determine which log events are noise or not noise. These methods will be Bag of Words Multinomial Naïve Bayes model, Multinomial Naïve Bayes with TF-IDF model and Multi-Variate Bernoulli Naïve Bayes model. Chapter IV will present all the findings from the models and lastly Chapter V will have the concluding remarks and future works.

CHAPTER II

LITERATURE REVIEW

In this chapter, the review of Naïve Bayesian classification and prediction models in literature.

2.1 Literature Review

2.1.1 Bayesian spam detection.

This paper was published in the Scholarly Horizons by the University of Minnesota [7]. It discusses the use of Bayesian filters to detect spam emails and looks at optimizations of Naïve Bayes text classification. It starts off by stating that Bayesian text classification has been used for decades and it still relevant today. Bayesian classification methods are most commonly used in email spam filtering and can be used to classify many kinds of documents. Three methods of Bayesian classification were explored, those being Naïve Bayes, Multinomial Bayes and Multi-Variate Bayes.

Naïve Bayes spam detection is a common model to detect spam but it is mostly used with an optimization. Naïve Bayes must be trained with controlled data that is already defined as spam or ham so the model can be applied to real world situations. Naïve Bayes also assumes that the features that it is classifying such as individual words in an email are independent from each other. A filter is created using training data that has been labeled ham or spam. Then the model signs the probability that each feature is in spam. The probabilities are written as values between

0 and 1. The filter will then evaluate whether or not an email is spam based on the individual probabilities of each word.

Multinomial Naïve Bayes is an optimization that is used to make the Naïve Bayes filter more accurate. The same steps are performed as in the Naïve Bayes but this time the number of occurrences of each word are kept track of as a multiset of words and the probability of each word being spam or ham is determined. A smoothing parameter is applied to prevent the values from being 0 if there are few occurrences of a particular word. This prevents the calculation from being highly inaccurate and having to divide by zero errors. The resulting probability of the email being spam is then compared to the probability of the email being ham. The value that is the highest determines the prediction. Multinomial Bayes does not scale well to large documents where there is a higher likelihood of duplicate features. This increases calculation time.

Multi-Variate Naïve Bayes also known as Bernoulli Naïve Bayes is a method that is closely related to Multinomial Bayes. Like the multinomial approach it treats each feature individually but they are also treated as Booleans. Instead of using a multiset of words with counts. The multiset contains a value of 0 or 1 (true or false) for each word. We can find the probabilities of the parameters in a similar fashion to Multinomial Bayes, resulting in the following equation:

$$P(W|S) = \frac{1 + N_{W,S}}{2 + N_S}$$

$P(W|S)$ is the probability of a word being spam. $N_{W,S}$ is the total number of training spam documents that contain the word W. N_S is the total number of training documents that are classified as spam.

The advantages of Naïve Bayes are that it is a very simple algorithm that performs equally well against more complex classifiers. It also does not classify the email on the basis of one or two words but takes into account every single relevant word. Bayesian filtering can constantly adapt

to new forms of spam. As the filter identifying spam and ham, it adds that data to its filter and applies it to later emails. A disadvantage of the Bayesian filter is called Bayesian poisoning. This is when a spammer intentionally includes words or phrases that are specifically designed to trick a Bayesian filter into believing that the email is not spam.

The results were weighed by the accuracy given from the testing. Accuracy is measured based on precision and recall. Precision is the fraction of the documents classified as spam and are in fact spam. Recall is the fraction of total documents that are spam. The Multinomial Bayes performed the best of all the methods used in the tests and Multi-Variate performed the worst. Most importantly the authors observed that Bayesian methods of text classification are still relevant today and even when there are new and more complex methods available.

2.1.2 Identifying Valuable Information from Twitter During Natural Disasters

This paper was from the Proceedings of the American Society for Information Science and Technology in 2014 [10], where the authors proposed that social media is a vital source of information during a major event. They explain that with the exponential increase in volume of social media data, the increase in conversational data does not provide valuable information. They designed an effective set of features and used them as input to Naïve Bayes classifiers. A Bag of Words (BOW) feature set with over 3000 features was compared to their designed feature set containing only 8 features.

The approach they used was to develop a set of features for use in machine learning algorithms that would be able to accurately distinguish “informational” tweets from “conversational” ones. “Informational” tweets are defined as any tweets which would provide valuable concrete information to anybody viewing the tweet. “Conversational” tweets were defined as having no concrete information; the information would not be universally useful to anybody who could read

the tweet. 1086 tweets were manually labeled for the classification task and contained 139 informational and 943 conversational tweets.

In feature development a primary distinguishing factor between informational and conversational tweets is discrepancies that reveal formality. Aspects of formality include correct grammar, lack of slang, lack of swear words, etc. The most effective features in classification for this research will revolve around the idea of formality. The proposed features they looked at are URL extraction, Emoticons, Instructional keywords, Phone numbers, Internet slang, Retweets and Profanity. Sentence structure analysis was also used.

	Our Features	BOW	Combined (features + BOW)
Precision - conversational	0.916	0.928	0.939
Precision - informational	0.405	0.442	0.444
Recall - conversational	0.907	0.903	0.889
Recall - informational	0.435	0.522	0.609
F-Measure - conversational	0.912	0.916	0.913
F-Measure - informational	0.42	0.478	0.514
AUC	0.812	0.86	0.865

Table 2 Resulting data from classification techniques

Table 2 reports precision, recall and F-measure for each class. Also reported is the area under the Receiver Operating Characteristic curve (AUC).

The results from the methodologies are shown in Table 2. As can be seen, the combined "bag of words" and the selected features set resulted in the most accurate prediction model. The designed feature set performs similarly and in some cases better than the "bag of Words" approach. The "bag of words" approach used 1488 features which is significantly larger compared to the 9 features used in the selected feature approach. The conclusion reached is that if computational resources are not a concern then the combined "bag of words" and the selected features approach

works best. If there is a requirement for using the least amount of computational resources, then using only the selected features approach is best.

2.1.3 A Comparison of Event Models for Naïve Bayes Text Classification

This paper was part of the "Workshop on Learning for Text Categorization" published in 1998. The paper starts with describing the Naïve Bayes classifier approach is the simplest of Bayesian classifier models and it assumes that all attributes of the examples are independent of each other give the context of the class. While this assumption is clearly false in most real-world tasks, Naïve Bayes often performs classification very well. Because of the independence assumption, the parameters for each attribute can be learned separately. This simplifies learning especially when the number of attributes is large. Document classification is a domain which contain a large number of attributes such as words. Naïve Bayes has been successfully applied to document classification in many research efforts. The article explains two models, Multi-Variate Bernoulli and Multinomial, with the Multi-Variate Bernoulli model sometimes outperforming the multinomial at small vocabulary sizes. Overall though the multinomial model provides a 27% reduction in error over the Multi-Variate Bernoulli, on average across all data sets.

In the Multi-Variate Bernoulli event model, a document is a binary vector over the space of words. Given a vocabulary, each dimension of the space corresponds to a word from the vocabulary. The dimension of the vector for the document is either a 1 or a 0. This indicates whether a word occurs at least once in the document. With such a document representation the Naïve Bayes assumption that the probability of each word occurring in a document is independent of the occurrence of other words in a document. Then the probability of a document given its class is simply the product of the probability of the attribute values over all word attributes.

The multinomial model captures word frequency information in documents. A document is an ordered sequence of word events, drawn from the same vocabulary. It is assumed that the lengths

of documents are independent of class. It is also assumed that the probability of each word event in a document is independent of the word's context and positon in the document. This model uses a bag of words representation of the documents where the number of times a word occurs in a document is counted. Thus the probability of a document given its class is simply the multinomial distribution. All the parameters of these models can be obtained by using training documents. Classification can then be determined by calculating the posterior probability of each class given the evidence of the test documents and selecting the class with the highest probability.

The empirical evidence documented in this article showed that the multinomial event model usually performed better than the Multi-Variate Bernoulli. The results are based on five different datasets. They are "Yahoo! Science" consisting of 13,589 pages; Market Guide Inc. consisting of 6,440 web pages; Newsgroups consisting of about 20,000 articles; WebKB 4,199 webpages; and ModApte 12,902 newswire articles.

The results show that Multi-Variate Bernoulli handles large vocabularies poorly and the multinomial event model is more appropriate for classifying documents with large vocabularies. Future work would be to show that multinomial event model should be more accurate classifier for data sets that have a large variance in document length. The multinomial event model handles documents of varying length by incorporating the evidence of each appearing work. The Multi-Variate Bernoulli model is a poor fit for data with varying length; it is more likely for a word to occur in a long document regardless of the class.

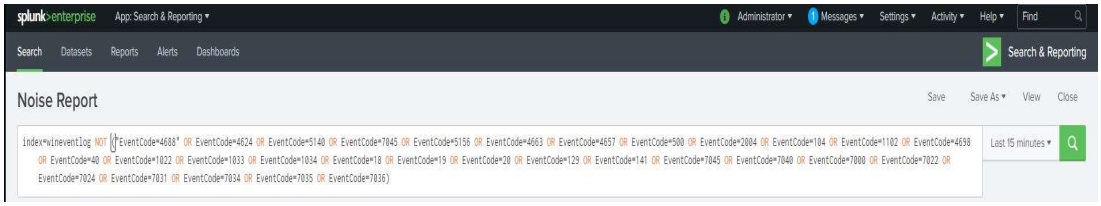
CHAPTER III

METHODOLOGY

This chapter looks at the method in preparing the training and testing data by running different Naïve Bayes methods to test classifying Windows event logs as either noise or non-noise. It will start with a description of the methods used and examples of Python Pseudo code demonstrating how those models are coded using the Scikit-learn library. This research will look for the model with the highest accuracy with the least number of feature words needed. The precision, recall and F1 will be measured for each model.

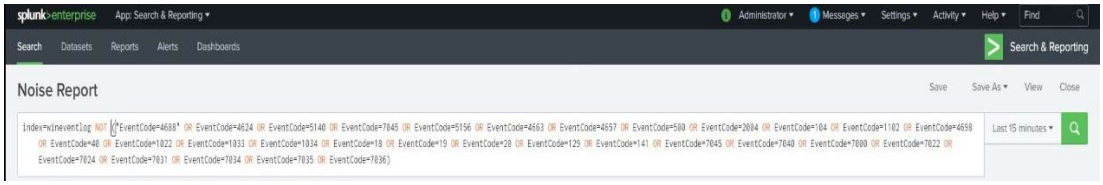
3.1 Data Collection

Data used in this research was collected from a large global company with any information which would identify that company has been obscured. The log events chosen are Windows log events that are from application, security and audit Window server logs. Windows log events contain event classification numbers in their text. A list of log event numbers was chosen (these can be referenced in the Appendices) which represented the possibility of malicious activity happening on a server. Searches were performed using Splunk SIEM software. These searches returned results containing log events which would represent noise or non-noise (possible malicious activity). They were exported to text files. Examples of the searches are shown in following figures.



index=wineventlog "EventCode=4688" OR EventCode=4624 OR EventCode=5140 OR EventCode=7045 OR EventCode=5156 OR EventCode=4663 OR EventCode=4657 OR EventCode=500 OR EventCode=2004 OR EventCode=104 OR EventCode=1102 OR EventCode=4698 OR EventCode=40 OR EventCode=1022 OR EventCode=1033 OR EventCode=1034 OR EventCode=18 OR EventCode=19 OR EventCode=20 OR EventCode=129 OR EventCode=141 OR EventCode=7045 OR EventCode=7040 OR EventCode=7000 OR EventCode=7022 OR EventCode=7024 OR EventCode=7031 OR EventCode=7034 OR EventCode=7035 OR EventCode=7036

Figure 3 Search Condition for Windows Non-Noise Log Events Using Splunk SIEM and Text Example of the Search



index=wineventlog NOT ("EventCode=4688" OR EventCode=4624 OR EventCode=5140 OR EventCode=7045 OR EventCode=5156 OR EventCode=4663 OR EventCode=4657 OR EventCode=500 OR EventCode=2004 OR EventCode=104 OR EventCode=1102 OR EventCode=4698 OR EventCode=40 OR EventCode=1022 OR EventCode=1033 OR EventCode=1034 OR EventCode=18 OR EventCode=19 OR EventCode=20 OR EventCode=129 OR EventCode=141 OR EventCode=7045 OR EventCode=7040 OR EventCode=7000 OR EventCode=7022 OR EventCode=7024 OR EventCode=7031 OR EventCode=7034 OR EventCode=7035 OR EventCode=7036)

Figure 4 Search Condition for Windows Noise Log Events Using Splunk SIEM and Text Example of the Search

3.2 Data Cleansing Event Logs for Text Classification

Before event logs can be processed by the Naïve Bayes algorithms they must be cleansed of anything that is not a word. This involves stripping out punctuation, symbols and numbers. Then

the event logs must be tokenized and stripped of stop words. Tokenization and normalization are the general processes of breaking down a text document into individual elements or words. These will serve as input for the Naïve Bayes natural language processing algorithms. Stop words are words that are particularly common in a set of text documents and are considered uninformative or having no value to the algorithm. The most common words in text documents are articles, prepositions, and pronouns, etc. Words like "of, are, the, it, is" are some examples of stop words. They are generally used so frequently that they start losing their semantic meaning. Removing stop words reduces the dimensionality of term space.[12] One approach to stop word removal is to search against a language-specific stop word dictionary. Another method is to create a stop list by sorting all words in the entire set of text documents by frequency. The stop list is converted into a set of non-redundant words and then is used to remove all those words from the input documents that are ranked among the top number of words in this stop list. Stop word removal in this research will be handled by the vectorization functions in the Scikit-learn Naïve Bayes model algorithms. Another step in text normalization is the process of stemming and lemmatization. This reduces inflectional forms and sometimes derivationally related forms of a word to a common base form. Since these are standardized Windows log events, stemming and lemmatizing along with grammar correction should not be needed.

Windows log events are text. For Naïve Bayes machine learning algorithms to be able to process this text the following steps need to be taken and were written into the algorithm used in this projects research.

1. Decoding Unicode characters into a normalized form, such as UTF8.
2. Strip Numerals.
3. Strip time prefixes such as "AM" and "PM".
4. Remove Punctuation.
5. Tokenize words and split by Whitespace.

6. Strip tabs, newlines and carriage returns and replace with blanks.

7. Strip multiple blanks and replace with a single blank.

```
for event in codecs.open('c:/event_files/window_event_non_noise.txt', encoding="utf-8",
errors="replace").readlines():                    #Read file and convert to utf-8 unicode.
    event = re.sub(r'\d+', '', event)    event = re.sub(r'\d+', '', eventt)  #strip numerals
    event = re.sub(r"[=\- \" \\\\ <>/():;.%^&,*?${}_']", " ", eventt)  #strip punctuation
    event.replace('PM', ' ')                                              #strip time element PM
    event.replace('AM', ' ')                                              #strip time element AM
    event = re.sub(r'\s+', ' ', event)                                  #strips \t\n\r\f\v and blanks
```

Figure 5 Python Pseudo Code Example

3.3 Feature Selection

Features in Naïve Bayes models are unique, measurable attributes or properties for each observation or data point in a dataset. Features are usually numeric in nature and can be absolute numeric values or categorical features that can be encoded as binary features for each category in the list using a process called one-hot encoding. The process of extracting and selecting features is called feature extraction or feature engineering. The extracted features are fed into Naïve Bayes machine learning algorithms for learning patterns that can be applied on future new data points and calculating classification insights. These algorithms usually expect features in the form of numeric vectors because each algorithm is at heart a mathematical operation of optimization and minimizing loss and error when it tries to learn patterns from data points and observations. So, with textual data there is the added challenge of figuring out how to transform textual data and extract numeric features from it. The Vector Space Model is used to represent the feature set in this research. This concept and model is very useful when working with textual data and is very popular in information retrieval and document ranking. The Vector Space Model, also known as the Term Vector Model, is defined as a mathematical and algebraic model for transforming and representing text documents as numeric vectors of specific terms that form the

vector dimensions. Mathematically this can be defined as follows. Say we have a document D in a document vector space VS. The number of dimensions or columns for each document will be the total number of distinct terms or words for all documents in the vector space. So, the vector space can be denoted

$$VS = \{ W, W, Wn \}$$

where there are n distinct words across all documents. Now we can represent document D in this vector space as

$$D = \{ wD_1, wD_2, wD_3, wD_n \}$$

were wDn denotes the weight for word n in document D. This weight is a numeric value and can represent anything ranging from the frequency of that word in the document, to the average frequency of occurrences or even to the TF-IDF weight. [8]

3.4 Naïve Bayes Classifiers Overview

Naïve Bayes classifiers are linear classifiers that are known for being simple yet very efficient. The probabilistic model of Naïve Bayes classifiers is based on Bayes' theorem, and the adjective Naïve comes from the assumption that the features in a dataset are mutually independent. In practice, the independence assumption is often violated, but Naïve Bayes classifiers still tend to perform very well under this unrealistic assumption [9].

To understand how Naïve Bayes classifiers work, I will go over the concept of Bayes' rule. The probability model that was formulated by Thomas Bayes (1701-1761) is simple and powerful. It is as follows:

$$\text{posterior probability} = \frac{\text{conditional probability} \cdot \text{prior probability}}{\text{evidence}}$$

The posterior probability is one of the quantities involved in Bayes' rule. It is the conditional probability of a given event, computed after observing a second event whose conditional and unconditional probabilities were known in advance. The objective function in the Naïve Bayes probability is to maximize the posterior probability given the training data in order to formulate the decision rule. An assumption that Bayes classifiers make is that the samples are independent and identically distributed. They describe random variables that are independent from one another and are drawn from a similar probability distribution. Independence means that the probability of one observation does not affect the probability of another observation. An additional assumption of Naïve Bayes classifiers is the conditional independence of features. Under this Naïve assumption, the class-conditional probabilities or likelihoods of the samples can be directly estimated from the training data instead of evaluating all possibilities of the data. The prior probability is introduced that can be interpreted as priori knowledge. In the context of pattern classification, the prior probabilities are also called class priors, which describe "the general probability of encountering a particular class." In the case of noise log event classification, the priors could be formulated as

P(noise) = "the probability that any new log event is noise"

and

P(malicious) = 1−P(noise)

If the prior probabilities are following a uniform distribution, the posterior probabilities will be entirely determined by the class-conditional probabilities and the evidence term. Since the evidence term is a constant, the decision rule will entirely depend on the class-conditional

probabilities. The evidence P(x) can be understood as the probability of encountering a particular pattern x independent from the class label. Naïve Bayes Classifiers produce very efficient model for text classification. [14]

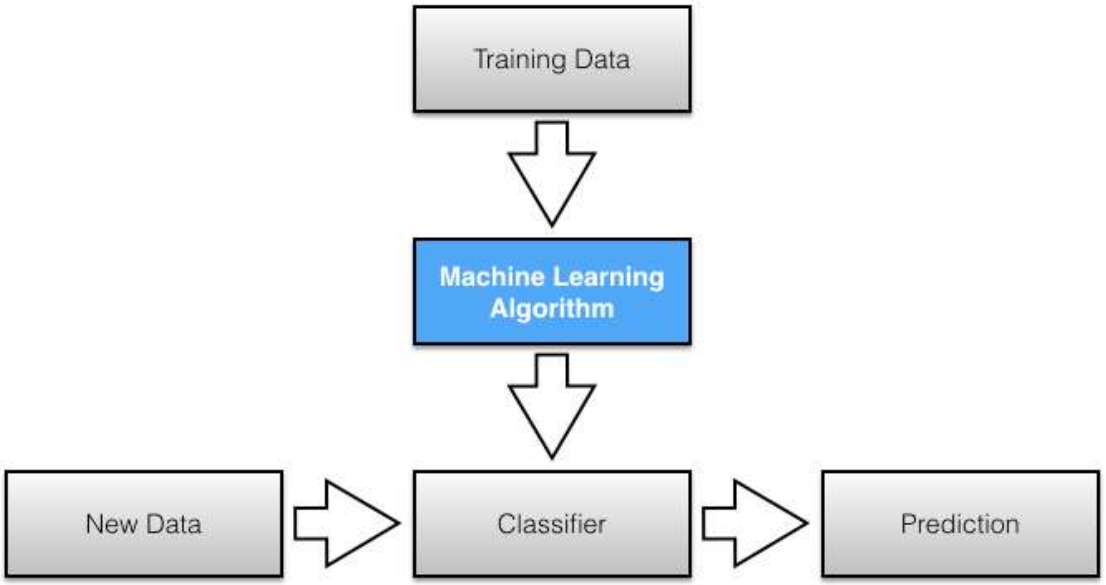


Figure 6 Diagram of Naïve Bayes Classifier [14]

3.5 Multinomial Naïve Bayes Overview

With a Naïve Bayes multinomial event model, feature vectors represent the frequencies with which certain events have been generated by a multinomial ($p_1, \dots, p_n$) where p_i is the probability that event i occurs. A feature vector $x = (x_1, \dots, x_n)$ is then a histogram, with x_i counting the number of times event i was observed in a particular instance. This is the event model typically used for document classification, with events representing the occurrence of a word in a single document. The likelihood of observing a histogram x is given by

$$p(\mathbf{x} \mid C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i {p_{ki}}^{x_i}$$

The multinomial Naïve Bayes classifier becomes a linear classifier when expressed in log-space.

$$\begin{aligned} \log p(C_k \mid \mathbf{x}) &\propto \log \left(p(C_k) \prod_{i=1}^{n} {p_{ki}}^{x_i} \right) \\ &= \log p(C_k) + \sum_{i=1}^{n} x_i \cdot \log p_{ki} \\ &= b + \mathbf{w}_k^\top \mathbf{x} \end{aligned}$$

where $b = \log p(C_k)$ and $w_{ki} = \log p_{ki}$.

If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero. This is problematic because it will wipe out all information in the other probabilities when they are multiplied. Therefore, it is often desirable to incorporate a small-sample correction, called pseudo count, in all probability estimates such that no probability is ever set to be exactly zero. This way of regularizing Naïve Bayes is called Laplace smoothing when the pseudo count is one, and Lidstone smoothing in the general case.

3.6 Multi-Variate Bernoulli Naïve Bayes Overview

In the Multi-Variate Bernoulli event model, features are independent Booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence features are used rather than term frequencies. If x_i is a Boolean expressing the occurrence or absence of the i'th term from the vocabulary, then the likelihood of a document given a class C_k is given by

$$p(\mathbf{x} \mid C_k) = \prod_{i=1}^{n} p_{ki}^{x_i} (1 - p_{ki})^{(1-x_i)}$$

where p_{ki} is the probability of class C_k generating the term x_i. This event model is especially popular for classifying short texts. It has the benefit of explicitly modeling the absence of terms.

3.7 Naïve Bayes Bag of Words Multinomial Model

The Bag of Words model is perhaps one of the simplest yet most powerful techniques to extract features from text documents. The essence of this model is to convert text documents into a vocabulary. This vocabulary can be understood as a set of non-redundant items where the order doesn't matter. A vector is then created which represents the count of all the distinct words that are present in the vocabulary for that specific document. The term Multinomial Naïve Bayes is a specific instance where each probability of observed features is a multinomial distribution, rather than some other distribution. This works well for data which can easily be turned into counts, such as word counts in text.

```
pipe = Pipeline(
        ['vectorBOW',  CountVectorizer(max_features=max,stop_words='english',
        lowercase=True)),
        ('clf',   MultinomialNB(alpha=1))])

predicted_fit = pipe.fit(train,train_label)

# predicted_test = Prediction Vector of Testing Data #
predicted_test = pipe.predict(test)


#  metric_Precision_Confidence = Precision or Confidence Factor   #

metric_Precision_Confidence =round(metrics.accuracy_score(test_label,
predicted_test),4)
```

Figure 7 Python Pseudo Code Example Multinomial Bag of Words Classifier

3.8 Naïve Bayes TF-IDF Multinomial Model

This model creates a vocabulary, a collection of all the different words that occur in the log event text sets and each word is associated with a Term Frequency - Inverse Document Frequency (TF-IDF). The TF-IDF approach assumes that the importance of a word is inversely proportional to how often it occurs across all text sets. TF-IDF is especially useful if stop words have not been removed from the text set. The goal of using TF-IDF is to scale down the impact of words that occur very frequently in a given text sets and that are hence empirically less informative than features that occur in a small fraction of the training text sets.

```
pipe = Pipeline([
    ('vect',  CountVectorizer(max_features=max,stop_words='english',
lowercase=True)),
    ('tfid',  TfidfTransformer(use_idf=True, smooth_idf=True)),
    ('clf',   MultinomialNB(alpha=1))])

predicted1 = pipe.fit(train,train_label)
predicted2 = pipe.predict(test)
metric =round(metrics.accuracy_score(test_label, predicted2),4)
```

Figure 8 Python Pseudo Code Example Multinomial TF-IDF Classifier

3.9 Naïve Bayes Multi-Variate Bernoulli Model

The Multi-Variate Bernoulli model is based on binary data: Every word in the vocabulary feature vector of the text sets is associated with the value 1 or 0. The feature vector has m dimensions where m is the number of words in the whole text set; the value 1 means that the word occurs in the particular text set, and 0 means that the word does not occur in this text set.

```
#Bernoulli Naïve Bayes Classifer
pipe_ber = Pipeline([
    ('vect', CountVectorizer(max_features=max, stop_words='english',
lowercase=True)),
    ('tfidf', TfidfTransformer(use_idf=True, smooth_idf=True)),
    ('clf', BernoulliNB()),])

pipe_ber.fit(train,train_label)
pred5 = pipe_ber.predict(test)
metric1 =round(metrics.accuracy_score(test_label, pred5),4)
print('BernoulliNB Classifier  ')
```

Figure 9 Python Pseudo Code Example Multi-Variate Bernoulli Classifier

3.10 Python Programming Library Used: Scikit-Learn

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. The scikit-learn project started as scikits learn, a Google Summer of Code project by David Cournapeau.

The pipeline class in Scikit-Learn is a compound classifier. To make the vectorizer => transformer => classifier workflow easier the Python Scikit-learn method sklearn.pipeline.Pipeline(steps, memory=None, verbose=False) will be used.

Scikit-learn's pipeline class is a useful tool for encapsulating multiple different transformers alongside an estimator into one object, so that you only have to call the fit() and predict methods once.

3.11 Scikit-learn Methods Used

CountVectorizer(max_features=max, stop_words='english')

CountVectorizer converts a collection of text documents to a matrix of token counts. This implementation produces a sparse representation of the counts using scipy.sparse.csr_matrix.

stop_words : string {'english'}, list, or None (default). If 'english', a built-in stop word list for English is used.

max_features : int or None, default=None. [13]

TfidfTransformer(use_idf=True, smooth_idf=True))

Transforms a count matrix to a normalized tf or tf-idf representation. Tf means term-frequency while tf-idf means term-frequency times inverse document-frequency. This is a common term weighting scheme used in document classification. The goal of using tf-idf is that instead of the raw frequencies of occurrence of a word in a given document is to scale down the impact of words that occur very frequently in a given collection of documents. The formula that is used to compute the tf-idf for a term t of a document d in a document set is

$$\text{tf-idf}(t, d) = \text{tf}(t, d) * \text{idf}(t)$$

and the idf is computed as

$$\text{idf}(t) = \log [\ n / \text{df}(t)\ ] + 1 \text{ (if smooth\_idf=False)}$$

where n is the total number of documents in the document set and df(t) is the document frequency of t; the document frequency is the number of documents in the document set that contain the term t. The effect of adding "1" to the idf in the equation above is that terms with zero idf, i.e., terms that occur in all documents in a training set, will not be entirely ignored. (Note that the idf formula above differs from the standard textbook notation that defines the idf as

$$\text{idf}(t) = \log [\ n / (\text{df}(t) + 1)\ ])$$

If smooth_idf=True (the default), the constant "1" is added to the numerator and denominator of the idf;

$$\text{idf}(d, t) = \log [\ (1 + n) / (1 + \text{df}(d, t))\ ] + 1$$

as if an extra document was seen containing every term in the collection exactly once, which prevents zero divisions. [13]

MultinomialNB(alpha=1.0)

The multinomial Naïve Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. [13]

alpha : float, optional (default=1.0) Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing). [13]

BernoulliNB ()

Like MultinomialNB, this classifier is suitable for discrete data. The difference is that while MultinomialNB works with occurrence counts, BernoulliNB is designed for binary/boolean features. [13]

3.12 Summarization of Algorithm Steps

1. Noise log events.
    a. Export noise log events from Splunk software and save to text file.
    b. Run data cleansing routine on log event files.
    c. Save to pandas dataframe.
    d. Add second column to dataframe and add the value of “noise” to each row for that column.
2. Export non-noise log events from Splunk software and save to text file.
    a. Export non-noise log events from Splunk software and save to text file.
    b. Run data cleansing routine on log event files.
    c. Save to pandas dataframe.
    d. Add second column to dataframe and add the value of “non-noise” to each row for that column.

3. Perform the test_train_split method and save return values to train, test, train_label, test_label variables.

4. Perform the Pipeline algorithm on all three Naïve Bayes models with the max_features variable set to 1000 through 50.

5. Set up Pipeline: Multinomial Naïve Bayes Bag of Words
    a. Initialize CountVectorizer.
    b. Initialize Transformer
    c. Call fit method and pass value of "train_label".
    d. Call predict method and pass value of "test" variable and save results to variable.
    e. Generate a Precision/Accuracy value.
    f. Generate a Classification Report.
    g. Generate a Confusion Matrix.

6. Set up Pipeline: Multinomial Naïve TF-IDF

    a. Initialize CountVectorizer.
    b. Initialize Transformer
    c. Call fit method and pass value of "train_label".
    d. Call predict method and pass value of "test" variable and save results to variable.
    e. Generate a Precision/Accuracy value.
    f. Generate a Classification Report.
    g. Generate a Confusion Matrix.

7. Set up Pipeline BernoulliNB
    a. Initialize CountVectorizer.
    b. Initialize Transformer
    c. Call fit method and pass value of "train_label".
    d. Call predict method and pass value of "test" variable and save results to variable.
    e. Generate a Precision/Accuracy value.
    f. Generate a Classification Report.
    g. Generate a Confusion Matrix.

8. Compare Accuracy, Precision and Recall values. Determine the smallest feature word set has the highest prediction accuracy.

CHAPTER IV

FINDINGS

This paper has proposed testing Bag of Words, Multinomial and Bernoulli Naïve Bayes Models to predict which Windows Event logs can be classified as noise. The models were run using the same training and test data. The variable changed in the processing of each model was the maximum number of features. The range consisted of 1000, 500, 400, 300, 200 and 100 feature words.

This chapter presents the results obtained from those tests using the algorithm steps proposed at the end of chapter 3. The results are shown as tables representing the accuracy, precision, recall, F1 and a confusion matrix for each model were generated. Graphs comparing Precision vs. Recall across the feature test range was generated for each model as well as a graph plotting the accuracy for each model across the feature range.

The 1000 feature test was included as an outlier to show that the accuracy results do not increase very much as the feature word sets get larger. The accuracy difference between the 1000 feature word set and the 200 feature word set is just .02. Using feature sets smaller than 200 and the accuracy starts falling and the models start breaking down in the results generated.

.

Naïve Bayes Model Results.

Feature Values 1000								
MultinomialNB Bag of Words Classifier								
	Confusion Matrix			**Classification Report:**				
Accuracy or Confidence Factor	**TP**	**FP**			**Precision**	**Recall**	**F1-Score**	**Support**
0.7231	319305	174311		0	0.65	0.72	0.68	442346
	123041	457257		1	0.79	0.72	0.75	631568
	FN	**TN**						
True Positve Rate	0.72							
False Positive Rate	0.28							
MultinomialNB TFIDF Classifier								
	Confusion Matrix			**Classification Report:**				
Accuracy or Confidence Factor	**TP**	**FP**			**Precision**	**Recall**	**F1-Score**	**Support**
0.7317	304868	188748		0	0.62	0.75	0.68	404269
	99401	480897		1	0.83	0.72	0.77	6696545
	FN	**TN**						
True Positve Rate	0.75							
False Positive Rate	0.28							
BernoulliNB Classifier								
	Confusion Matrix			**Classification Report:**				
Accuracy or Confidence Factor	**TP**	**FP**			**Precision**	**Recall**	**F1-Score**	**Support**
0.7233	319953	173663		0	0.65	0.72	0.68	443412
	123459	56839		1	0.79	0.72	0.75	630502
	FN	**TN**						
True Positve Rate	0.72							
False Positive Rate	0.75							

Figure 10 1000 Features Data Table

Feature Values 500								
MultinomialNB Bag of Words Classifier								
	Confusion Matrix			**Classification Report:**				
Accuracy or Confidence Factor	**TP**	**FP**			**Precision**	**Recall**	**F1-Score**	**Support**
0.7154	316356	177464		0	0.64	0.71	0.67	44479
	128123	451971		1	0.78	0.72	0.75	629435
	FN	**TN**						
True Positve Rate	0.71							
False Positive Rate	0.28							
MultinomialNB TFIDF Classifier								
	Confusion Matrix			**Classification Report:**				
Accuracy or Confidence Factor	**TP**	**FP**			**Precision**	**Recall**	**F1-Score**	**Support**
0.7252	301935	191885		0	0.61	0.75	0.67	405163
	103228	476866		1	0.82	0.71	0.76	668751
	FN	**TN**						
True Positve Rate	0.75							
False Positive Rate	0.29							
BernoulliNB Classifier								
	Confusion Matrix			**Classification Report:**				
Accuracy or Confidence Factor	**TP**	**FP**			**Precision**	**Recall**	**F1-Score**	**Support**
0.7174	317614	176206		0	0.64	0.71	0.68	444917
	127303	452791		1	0.78	0.72	0.75	628997
	FN	**TN**						
True Positve Rate	0.71							
False Positive Rate	0.28							

Figure 11 500 Features Data Table

Feature Values 400								
MultinomialNB Bag of Words Classifier								
	Confusion Matrix			**Classification Report:**				
Accuracy or Confidence Factor	**TP**	**FP**			**Precision**	**Recall**	**F1-Score**	**Support**
0.7126	315210	177528		0	0.64	0.71	0.67	446272
	131062	450114		1	0.77	0.72	0.74	627642
	FN	**TN**						
True Positve Rate	0.71							
False Positive Rate	0.28							
MultinomialNB TFIDF Classifier								
	Confusion Matrix			**Classification Report:**				
Accuracy or Confidence Factor	**TP**	**FP**			**Precision**	**Recall**	**F1-Score**	**Support**
0.722	301000	191738		0	0.61	0.74	0.67	407778
	106778	474398		1	0.82	0.71	0.76	666136
	FN	**TN**						
True Positve Rate	0.74							
False Positive Rate	0.29							
BernoulliNB Classifier								
	Confusion Matrix			**Classification Report:**				
Accuracy or Confidence Factor	**TP**	**FP**			**Precision**	**Recall**	**F1-Score**	**Support**
0.7136	316228	176510		0	0.64	0.71	0.67	447312
	131084	450092		1	0.77	0.72	0.75	626602
	FN	**TN**						
True Positve Rate	0.71							
False Positive Rate	0.28							

Figure 12 400 Features Data Table

Feature Values 300								
MultinomialNB Bag of Words Classifier								
	Confusion Matrix			**Classification Report:**				
Accuracy or Confidence Factor	**TP**	**FP**			**Precision**	**Recall**	**F1-Score**	**Support**
0.7147	300508	193630		0	0.61	0.73	0.66	413311
	112803	466973		1	0.81	0.71	0.75	660603
	FN	**TN**						
True Positve Rate	0.73							
False Positive Rate	0.29							
MultinomialNB TFIDF Classifier								
	Confusion Matrix			**Classification Report:**				
Accuracy or Confidence Factor	**TP**	**FP**			**Precision**	**Recall**	**F1-Score**	**Support**
0.7185	300179	193959		0	0.61	0.73	0.67	408540
	108361	471415		1	0.81	0.71	0.76	665374
	FN	**TN**						
True Positve Rate	0.73							
False Positive Rate	0.29							
BernoulliNB Classifier								
	Confusion Matrix			**Classification Report:**				
Accuracy or Confidence Factor	**TP**	**FP**			**Precision**	**Recall**	**F1-Score**	**Support**
0.7153	301024	193114		0	0.61	0.73	0.66	413653
	112629	467147		1	0.81	0.71	0.75	660261
	FN	**TN**						
True Positve Rate	0.73							
False Positive Rate	0.29							

Figure 13 300 Features Data Table

Feature Values 200

MultinomialNB Bag of Words Classifier

Accuracy or Confidence Factor	Confusion Matrix TP	Confusion Matrix FP		Classification Report:	Precision	Recall	F1-Score	Support
0.7074	294478	199233		0	0.6	0.72	0.65	409498
	115020	465183		1	0.8	0.7	0.75	664416
	FN	TN						
True Positve Rate	0.72							
False Positive Rate	0.30							

MultinomialNB TFIDF Classifier

Accuracy or Confidence Factor	Confusion Matrix TP	Confusion Matrix FP		Classification Report:	Precision	Recall	F1-Score	Support
0.7097	293096	200615		0	0.59	0.72	0.65	404286
	111190	469013		1	0.81	0.7	0.75	669628
	FN	TN						
True Positve Rate	0.72							
False Positive Rate	0.30							

BernoulliNB Classifier

Accuracy or Confidence Factor	Confusion Matrix TP	Confusion Matrix FP		Classification Report:	Precision	Recall	F1-Score	Support
0.7082	296851	196860		0	0.6	0.72	0.65	413361
	116510	463693		1	0.8	0.7	0.75	660553
	FN	TN						
True Positve Rate	0.72							
False Positive Rate	0.30							

Figure 14 200 Features Data Table

Feature Values 100

MultinomialNB Bag of Words Classifier

Accuracy or Confidence Factor	Confusion Matrix TP	Confusion Matrix FP		Classification Report:	Precision	Recall	F1-Score	Support
0.6872	281410	212064		0	0.57	0.69	0.63	405307
	123897	456543		1	0.79	0.68	0.73	668607
	FN	TN						
True Positve Rate	0.69							
False Positive Rate	0.32							

MultinomialNB TFIDF Classifier

Accuracy or Confidence Factor	Confusion Matrix TP	Confusion Matrix FP		Classification Report:	Precision	Recall	F1-Score	Support
0.6878	270361	223113		0	0.55	0.71	0.62	382558
	112197	468243		1	0.81	0.68	0.74	691356
	FN	TN						
True Positve Rate	0.71							
False Positive Rate	0.32							

BernoulliNB Classifier

Accuracy or Confidence Factor	Confusion Matrix TP	Confusion Matrix FP		Classification Report:	Precision	Recall	F1-Score	Support
0.684	265738	227736		0	0.54	0.7	0.61	377333
	111595	468845		1	0.81	0.67	0.73	696581
	FN	TN						
True Positve Rate	0.70							
False Positive Rate	0.33							

Figure 15 100 Features Data Table

Precision helps show how precise the model is indicating how many of the log events predicted positive are actual positive. It is also a good measure to determine the costs of false positives. The false positive is when a log event is predicted to be noise and is actually non-noise. When the precision is high then the false positives are low.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Recall actually calculates how many of the actual positives the models detect. The recall should be the metric used to select the best model when the cost associated with false negative predictions is high.

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

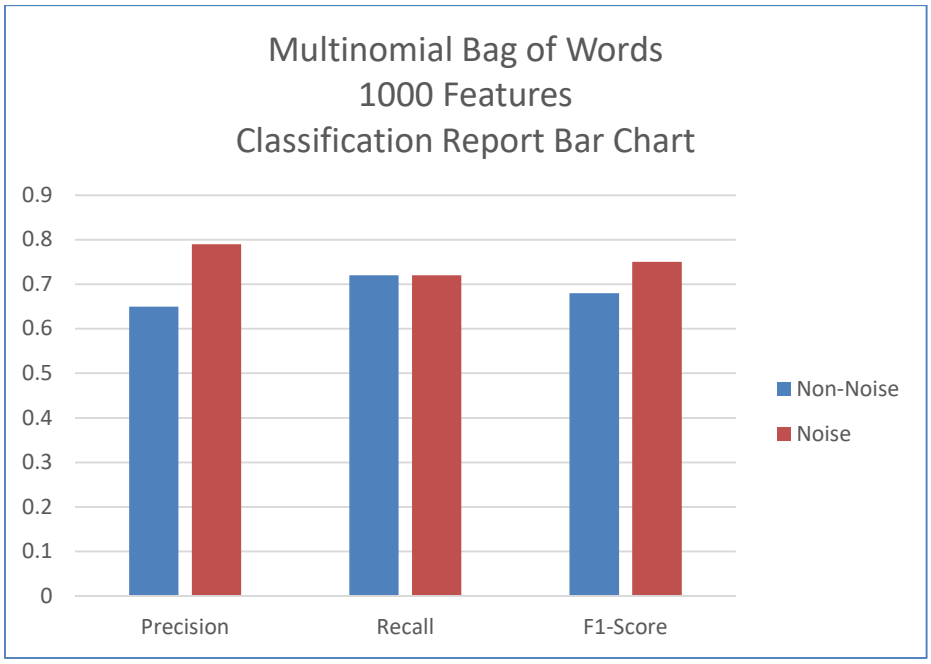


Figure 16 Multinomial Bag of Words 1000 Features Classification Report Bar Chart

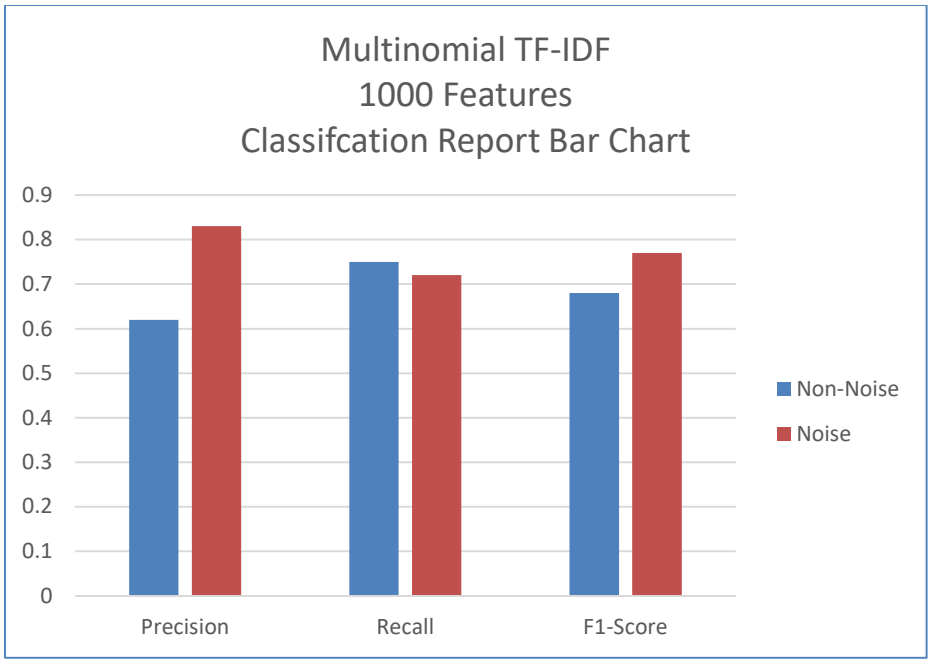


Figure 17 Multinomial TF-IDF 1000 Features Classification Report Bar Chart

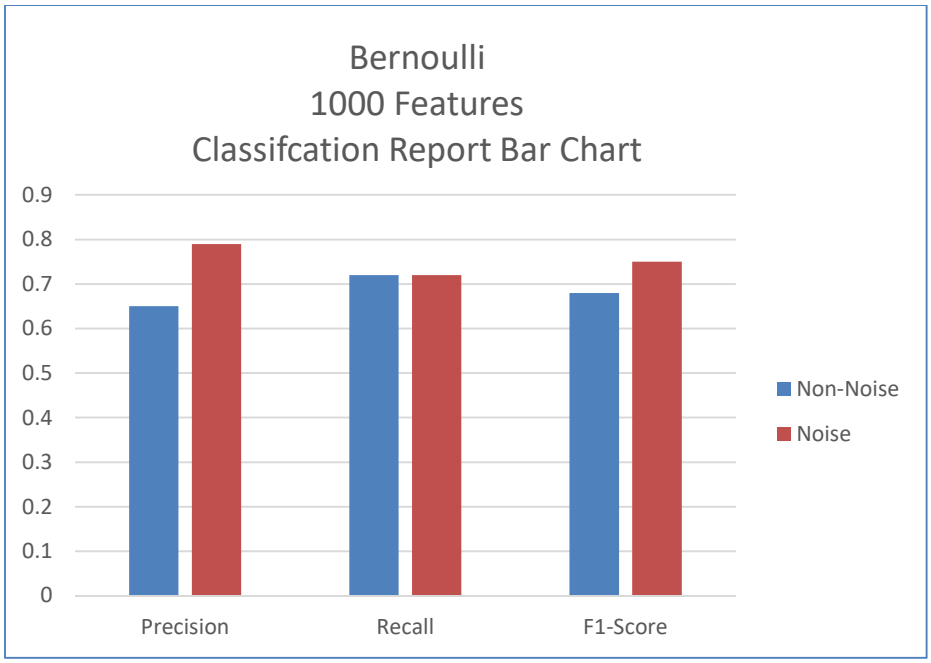


Figure 18 Bernoulli 1000 Features Classification Report Bar Chart

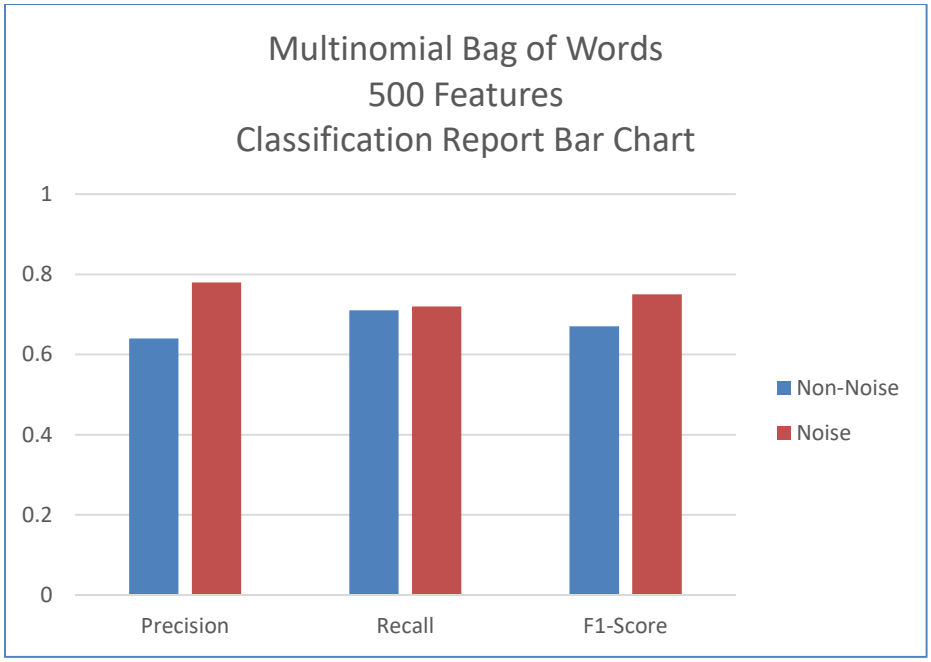


Figure 19 Multinomial Bag of Words 500 Features Classification Report Bar Chart

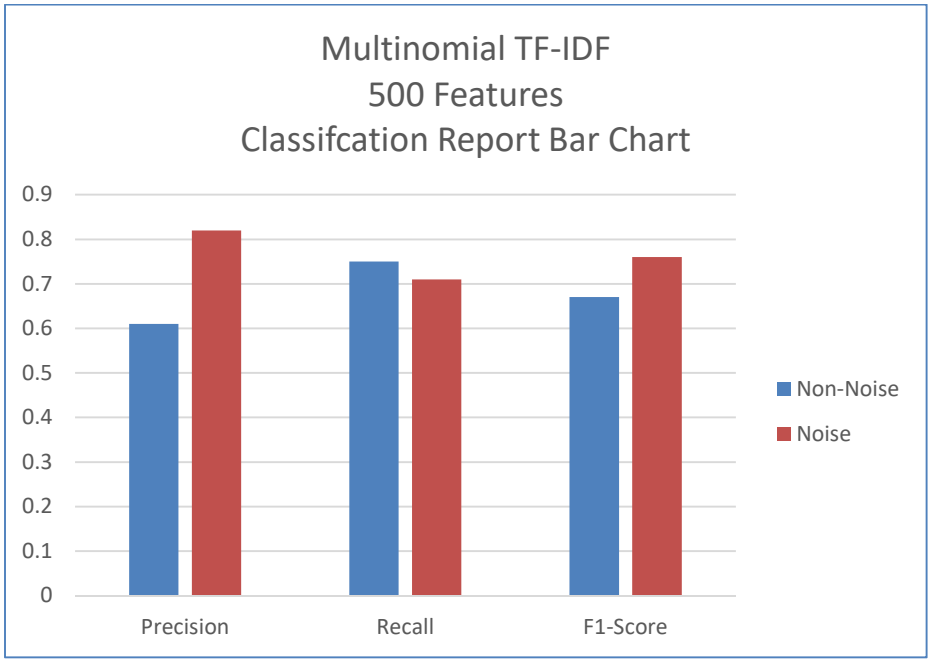


Figure 20 Multinomial TF-IDF 500 Features Classification Report Bar Chart

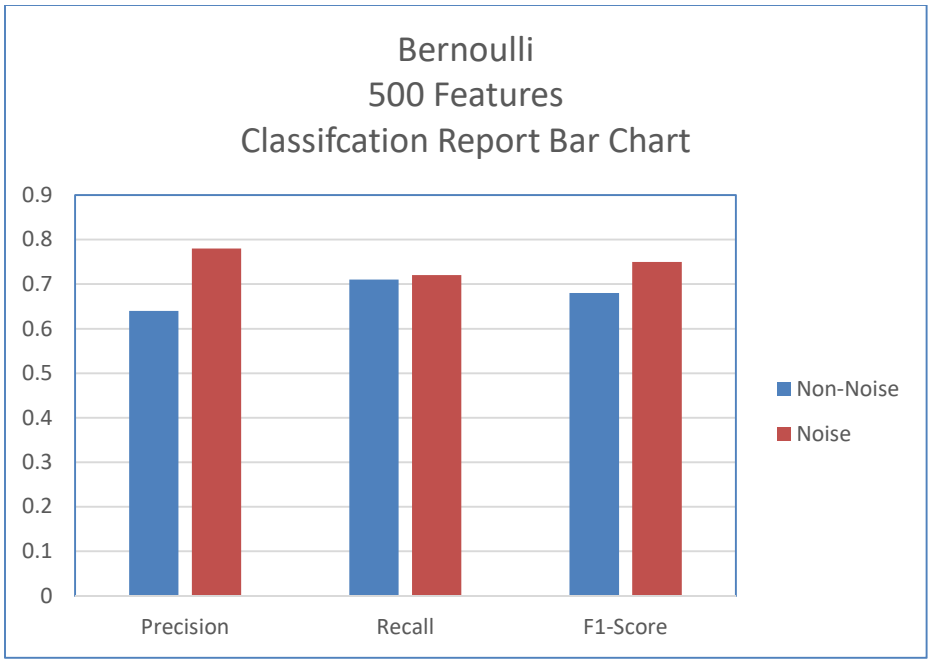


Figure 21 Bernoulli 500 Features Classification Report Bar Chart

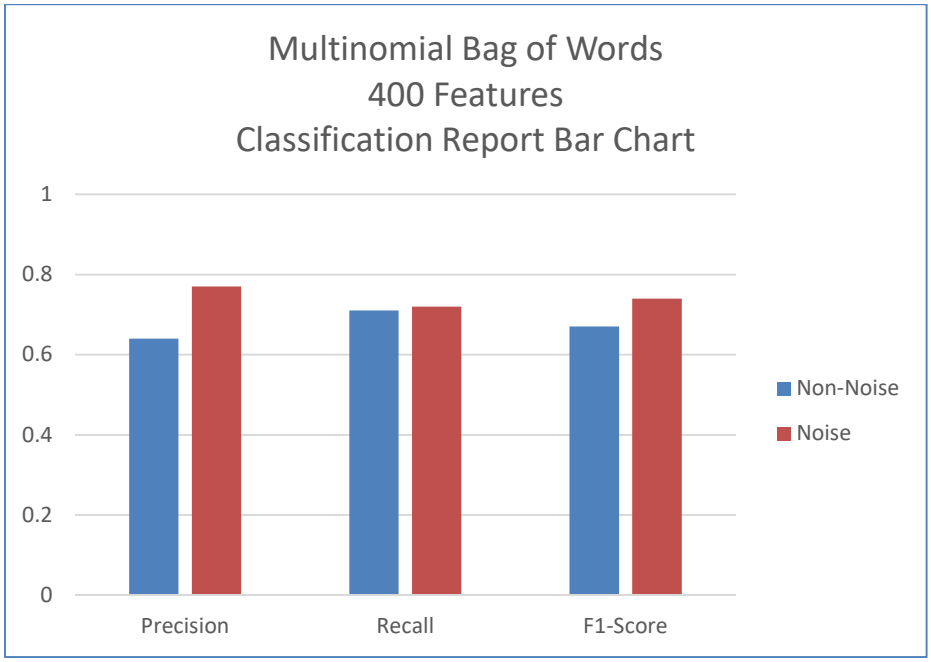


Figure 22 Multinomial Bag of Words 400 Features Classification Report Bar Chart

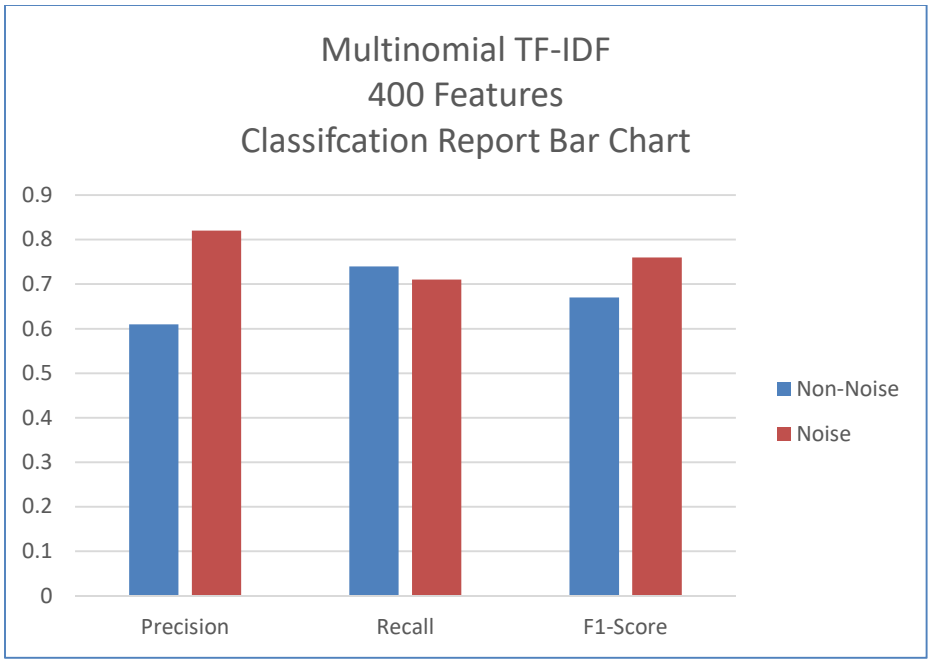


Figure 23 Multinomial TF-IDF 400 Features Classification Report Bar Chart

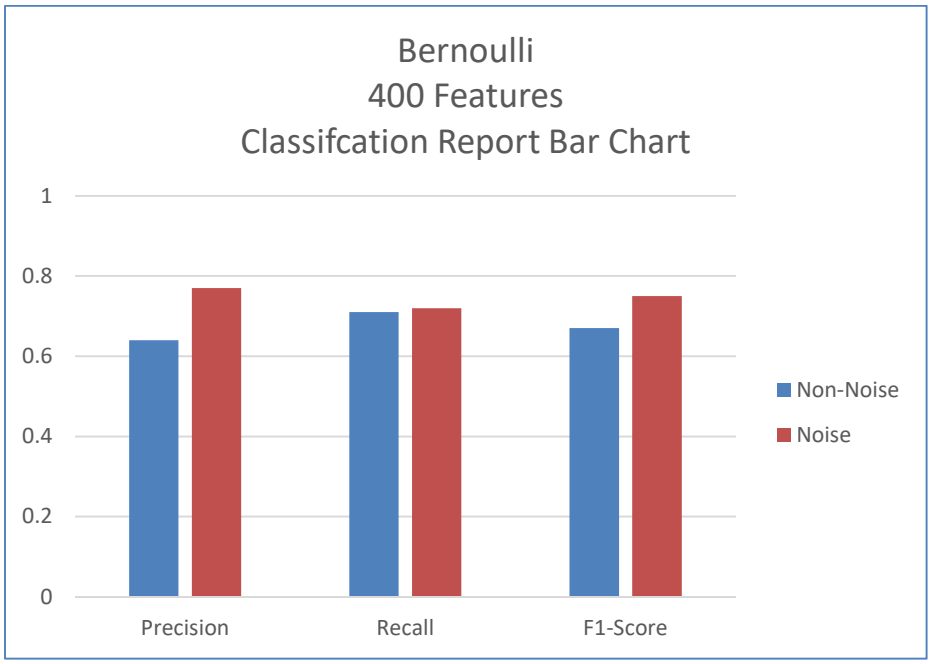


Figure 24 Bernoulli 400 Features Classification Report Bar Chart

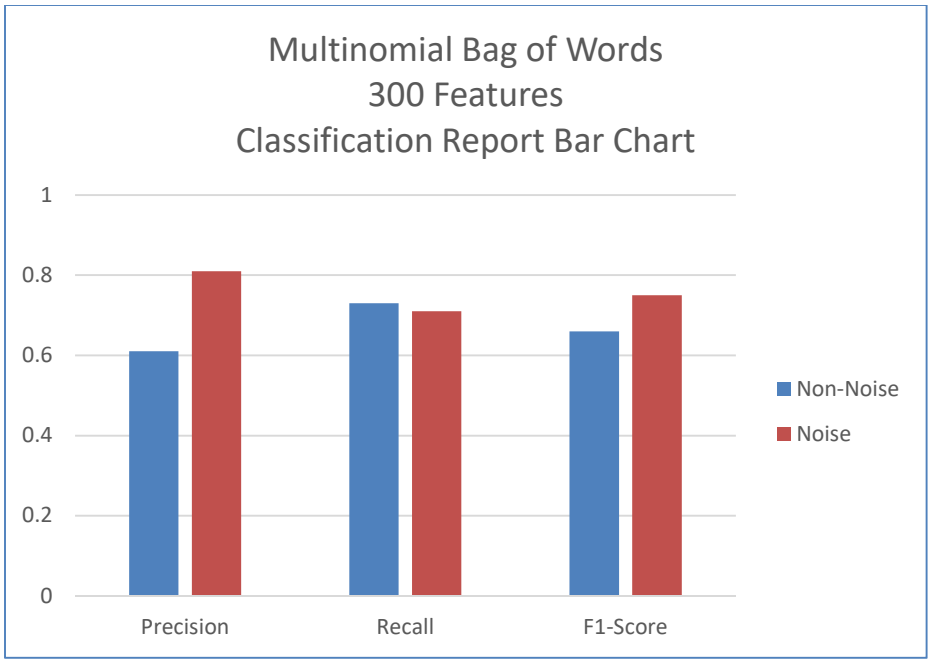


Figure 25 Multinomial Bag of Words 300Features Classification Report Bar Chart

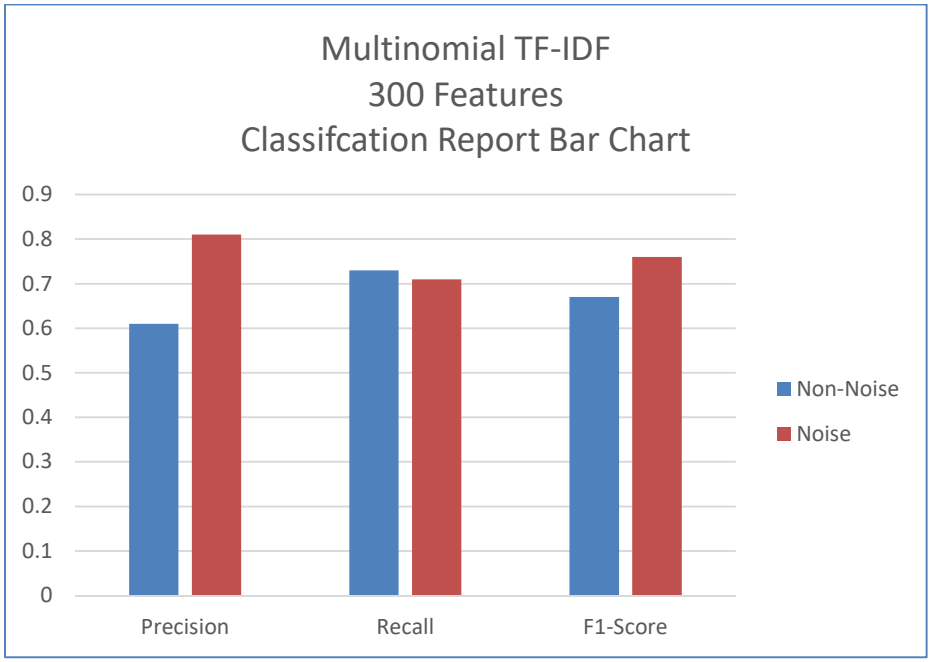


Figure 26 Multinomial TF-IDF 300 Features Classification Report Bar Chart

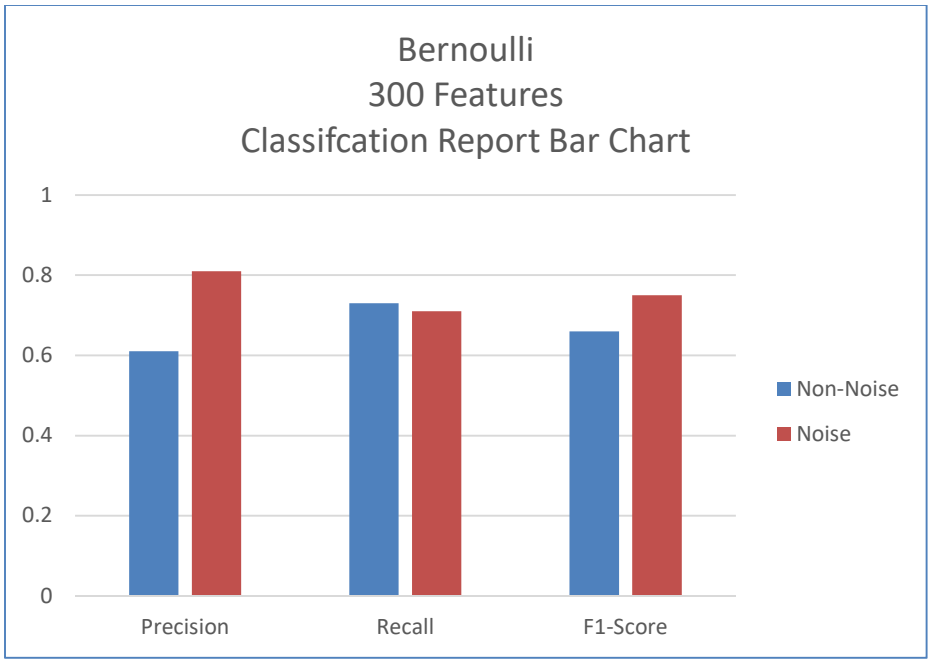


Figure 27 Bernoulli 300 Features Classification Report Bar Chart

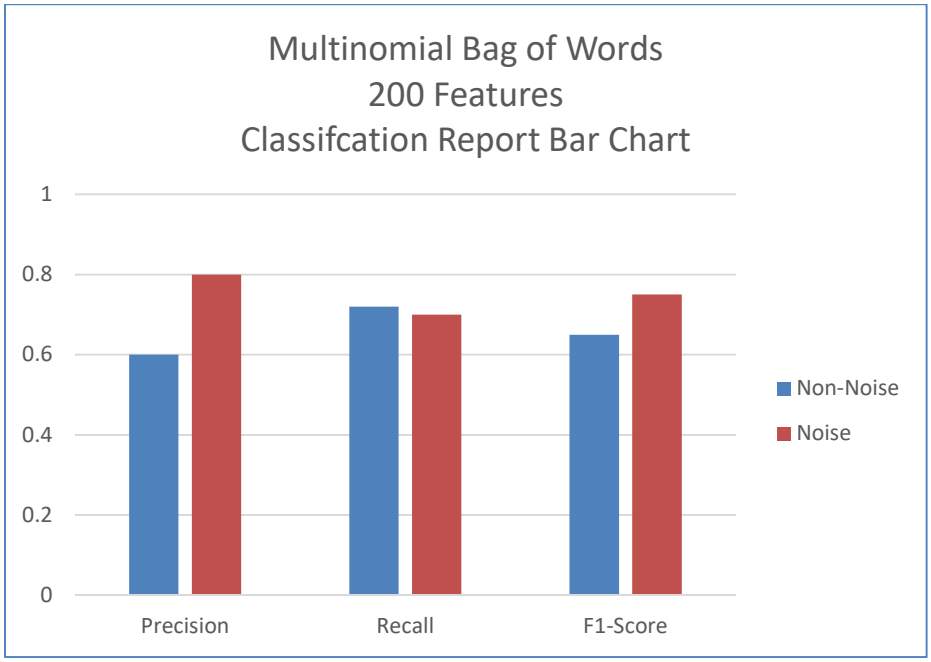


Figure 28 Multinomial Bag of Words 200 Features Classification Report Bar Chart

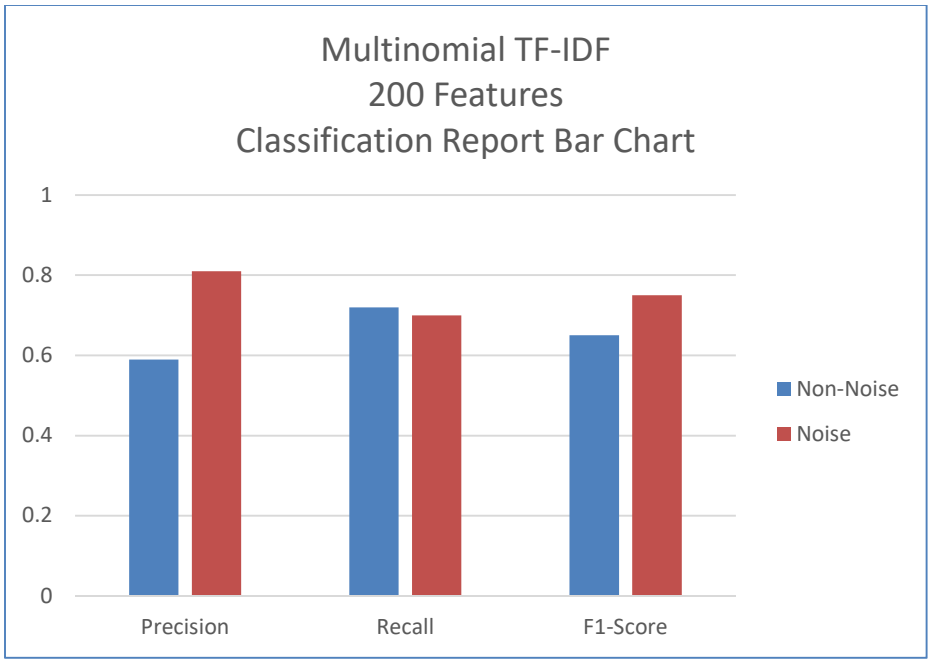


Figure 29 Multinomial TF-IDF 200 Features Classification Report Bar Chart

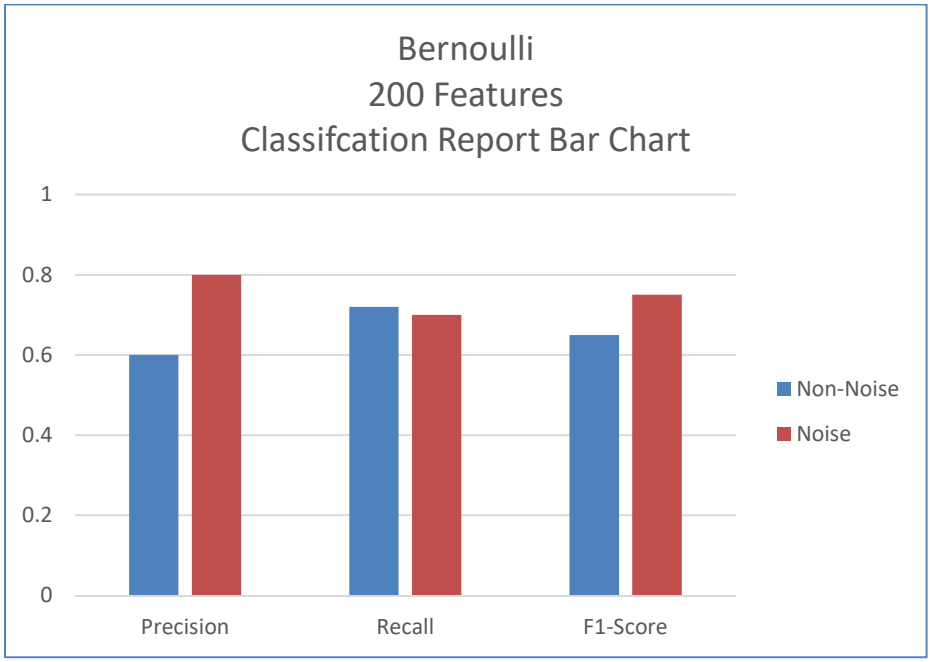


Figure 30 Bernoulli 200 Features Classification Report Bar Chart

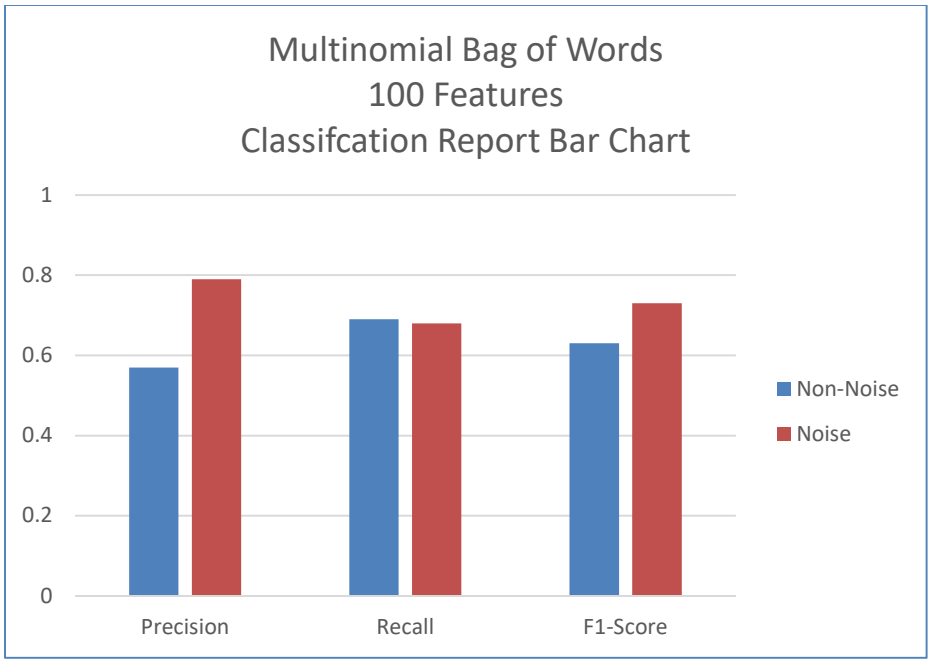


Figure 31 Multinomial Bag of Words 100 Features Classification Report Bar Chart

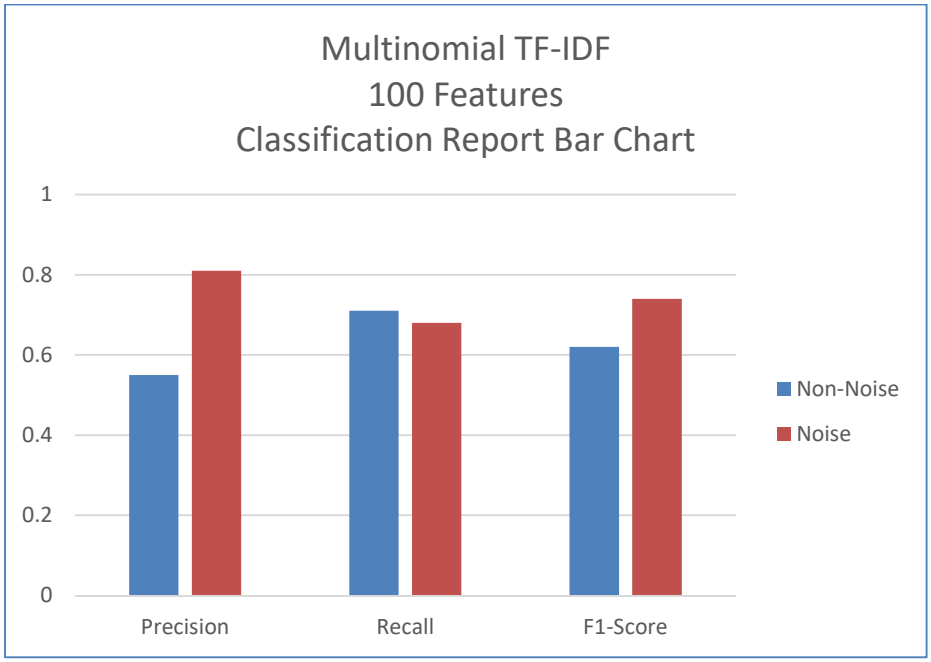


Figure 32 Multinomial TF-IDF 100 Features Classification Report Bar Chart

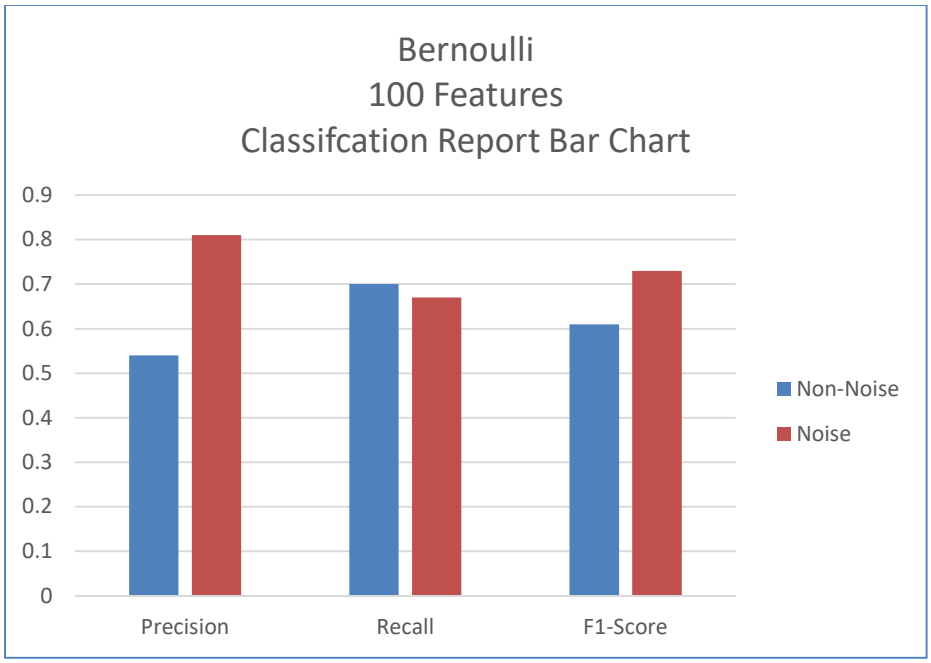


Figure 33 Bernoulli 100 Features Classification Report Bar Chart

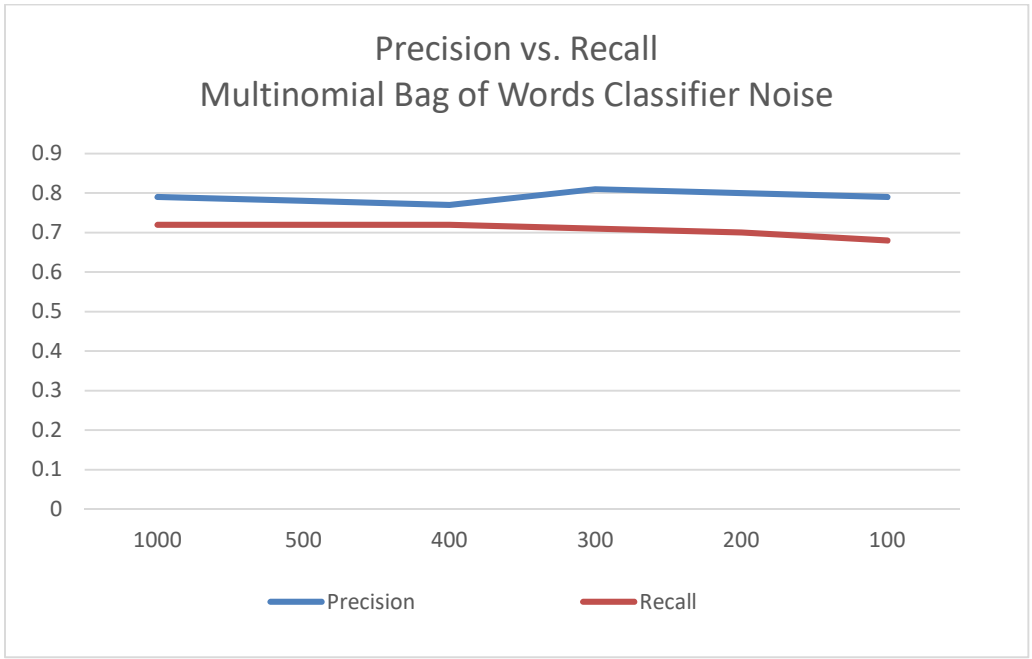


Figure 34 Precision vs. Recall Multinomial Bag of Words Classifier for Noise

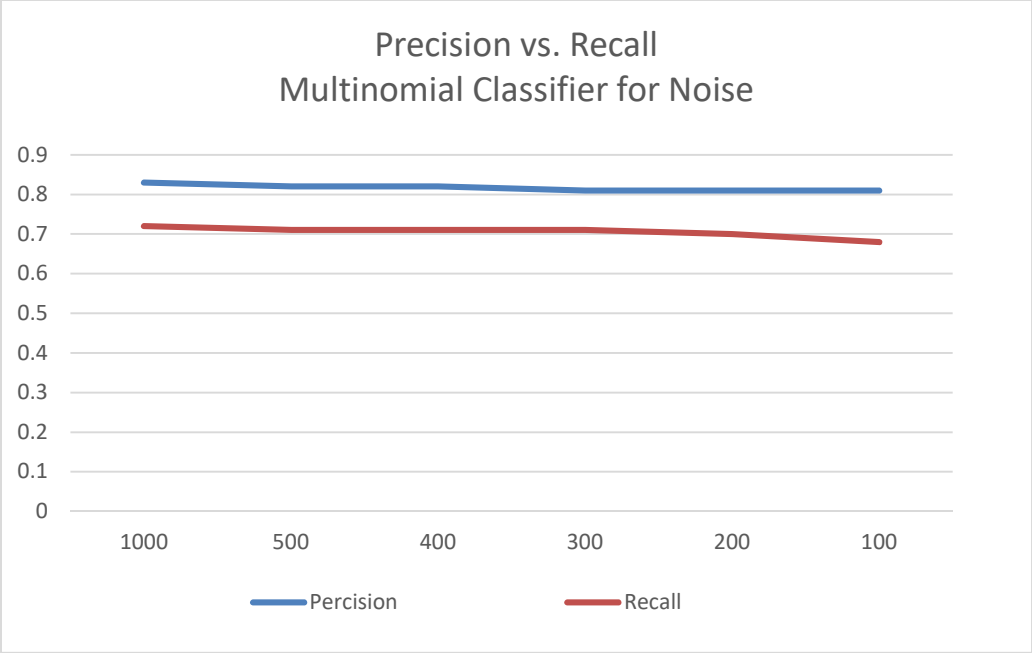


Figure 35 Precision vs. Recall Multinomial Classifier for Noise

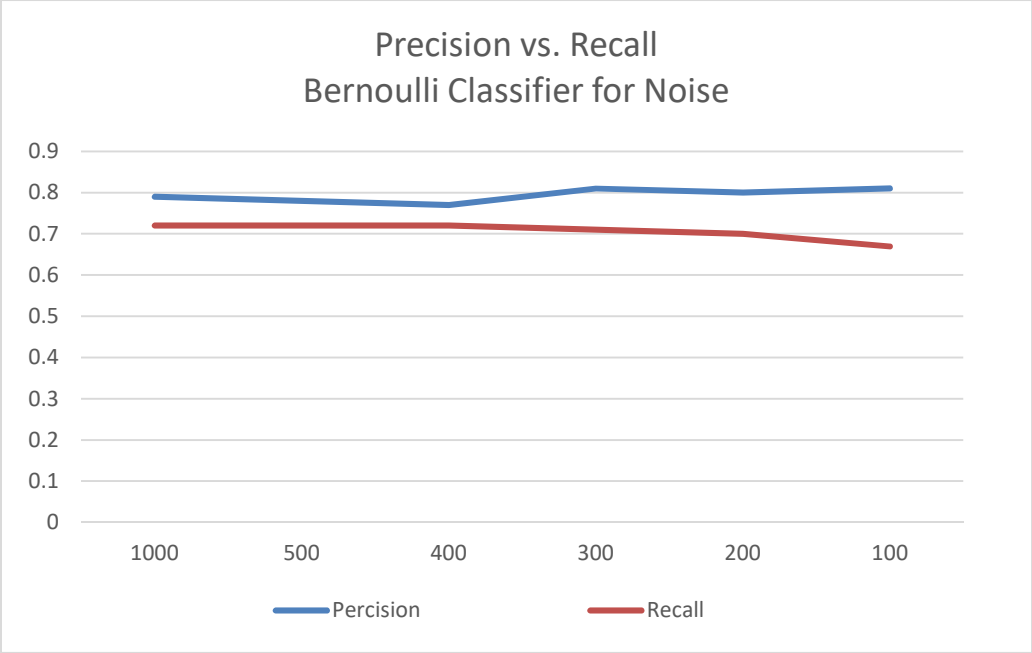


Figure 36 Precision vs Recall Bernoulli Classifier

Accuracy shows how well the model predicts the predicts whether a log event is noise or non-noise. There is not much difference between 1000 word features and 200 word features. Roughly a little over .02 points. The main observation is that the accuracy starts dropping when the feature word set is smaller 200 words. Even though the 200 feature word set in any of the models tested has the lowest accuracy in the tested range before the accuracy starts to decline, it is still only roughly .02 points lower than the 1000 feature word set.

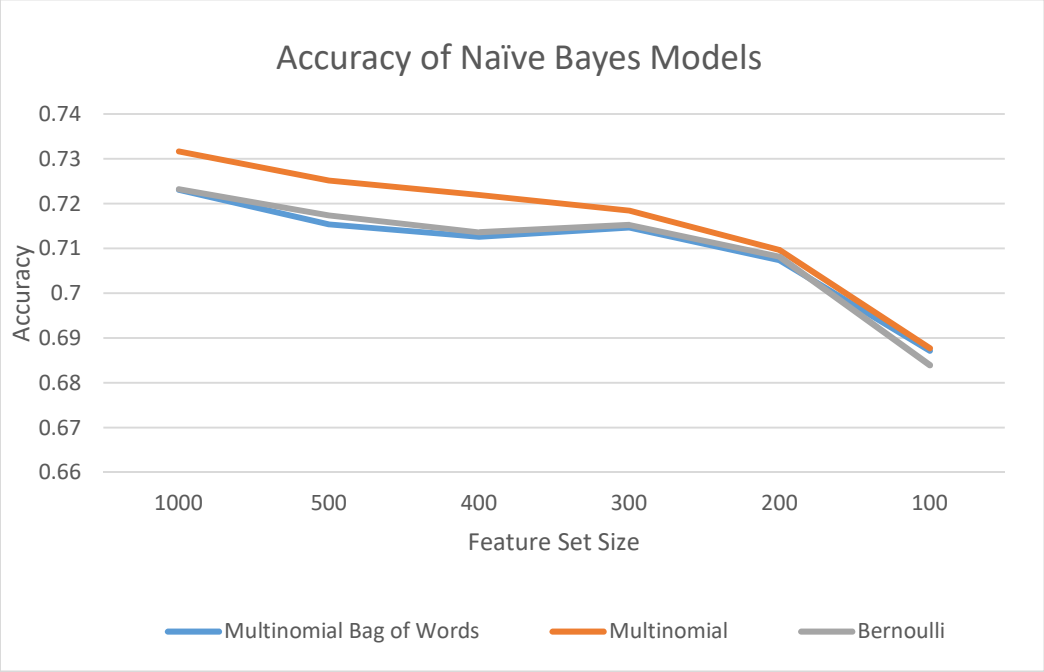


Figure 37 Accuracy of Naïve Bayes Models

CHAPTER V

CONCLUSION

This research is trying to determine the best Naïve Bayes model for predicting Noise and Non-Noise log events with the greatest accuracy score while using the feature set with the least number of words. All models show very similar precision and recall scores across all feature set sizes and the small space between the Precision and Recall lines shows a balance in the size of the test data used which should lead to better accuracy scores. The accuracy scores showed only a difference of .02 points between 1000 word feature set size and the 200 word feature set size. The main point of the Accuracy graph shows that all models start showing a reduced accuracy trend after 200 word feature set size. The small difference of the accuracy score at the 200 word feature set show that any of the models could be used for prediction of noise events in future works.

Possible future research would be to find a method of enhancement by manually selecting words contained in the feature set that would increase the accuracy score of the Naïve Bayes model predictions. Additional research should be performed to determine the average size reduction in the number of noise log events possible by filtering those log events through the Naïve Bayes models. This could be used to estimate the amount of resources that could be saved when the SIEM tools process flows use the Naïve Bayes predictors.

REFERENCES

[1] PCI Security Standards Council. "Payment Card Industry (PCI) Data Security Standard Requirements and Security Assessment Procedures Version 3.2.1 May 2018"

[2] United States. General Accounting Office. [*Information Security Comments on the Proposed Federal Information Security Management Act of 2002*. Washington, D.C.]: U.S. General Accounting Office, 2002

[3] IT Governance. Privacy Team, and European Union. EU General Data Protection Regulation (GDPR): An Implementation and Compliance Guide. Second ed. 2017. Web

[4] N.Y.COMP.CODES R.&REGS.tit.23, § 500.00 (2017)

[5] "Bill Summary & Status 106th Congress (1999–2000) S.900 CRS Summary – Thomas (Library of Congress)". Archived from the original on 2013-08-12. Retrieved 2011-02-08.

[6] Hall, James, and Stephen Liedtka. "The Sarbanes-Oxley Act." Association for Computing Machinery. Communications of the ACM 50.3 (2007): 95-100. Web.

[7] Eberhardt, J.J.: Bayesian Spam Detection. University of Minnesota, Morris Undergraduate Journal, Scholarly Horizons (2015)

[8] Sarkar, D.: Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Insights from Your Data. A Press, New York (2016)

[9] I. Rish, "An empirical study of the Naïve Bayes Classifier," in IJCAI 2001 workshop on empirical methods in artificial intelligence, pp. 41–46, 2001

[10] Truong, Brandon, Cornelia Caragea, Anna Squicciarini, and Andrea H. Tapia. "Identifying Valuable Information from Twitter during Natural Disasters." Proceedings of the American Society for Information Science and Technology 51.1 (2014): 1-4. Web

[11] Mccallum, Andrew & Nigam, Kamal. (2001). A Comparison of Event Models for Naïve Bayes Text Classification. Work Learn Text Categ. 752

[12] Elkafrawy, Passent & Khalaf, Mohamed. (2018). Forming System Requirements for Software Development Using Semantic Technology. 755-764. 10.1007/978-3-319-64861-3_71

[13] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011

[14] Raschka, Sebastian. (2014). Naïve Bayes and Text Classification I - Introduction and Theory. 10.13140/2.1.2018.3049

APPENDICES

Windows Events Which Show Possible Malicious Activity

Event Category: Event Code – Event Description

NEW PROCESS STARTING: Event Code 4688 - Will capture when a process or executable starts.

USER LOGON SUCCESS: Event Code 4624 - Will capture when a user successfully logons to the system.

SHARE ACCESSED: Event Code 5140 - Will capture when a user connects to a file share.

NEW SERVICE INSTALLED: Event Code 7045 will capture when a new service is installed.

NETWORK CONNECTION MADE: Event Code 5156 will capture when a network connection is made from the source to the destination including the ports used and the process used to initiate the connection. Requires the use of the Windows Firewall.

FILE AUDITING: Event Code 4663 will capture when a new file is added, modified or deleted.

REGISTRY AUDITING: Event Code 4657 will capture when a new registry item is added, modified or deleted.

WINDOWS POWERSHELL COMMAND LINE EXECUTION: Event Code 500 will capture when PowerShell is executed logging the command line used.

WINDOWS FIREWALL CHANGES: Event Code 2004 will capture when new firewall rules are added.

SCHEDULE TASKS ADDED: Event Code 4698: A scheduled task was created.

LOG CLEAR: Event Code 104 – SYSTEM Log – The “Windows PowerShell” or “PowerShell Operational” log was cleared.

LOG CLEAR: Event Code 1102 – SECURITY Log – The audit log was cleared.

TASKS: Event Code 4698 – SECURITY Log – New Task Created.

DRIVER: Event Code 40 – Issue with Driver.

INSTALLER: Event Code 1022 – Windows Installer – Updated the product.

INSTALLER: Event Code 1033 – Windows Installer – Installed the product.

INSTALLER: Event Code 1034 – Windows Installer – Removed the product.

WINDOWS UPDATE: Event Code 18 – Watch for the Windows Update Agent activity – Ready.

WINDOWS UPDATE: Event Code 19 – Watch for the Windows Update Agent activity – Installed.

WINDOWS UPDATE: Event Code 20 – Watch for the Windows Update Agent activity – Failure.

TASKSCHEDULER LOG: Event Code 129 - Watch for Created Task – Created.

TASKSCHEDULER LOG: Event Code 141 - Watch for Deleted Task – Deleted.

SERVICES: Event Code 7045 - A service was installed in the system.

SERVICES: Event Code 7040 - The start type of the XYZ service was changed from auto start to disabled.

SERVICES: Event Code 7000 - The XYX service failed to start due to the following error: The service did not respond to the start or control request in a timely fashion.

SERVICES: Event Code 7022 - The XYZ service hung on starting.

SERVICES: Event Code 7024 - The XYZ service terminated with service-specific error.

SERVICES: Event Code 7031 - The XYZ service terminated unexpectedly.

SERVICES: Event Code 7034 - The XYZ service terminated unexpectedly.

SERVICES: Event Code 7035 – Service sent a request to Stop or Start

SERVICES: Event Code 7036 – Service was Started or Stopped

VITA

Marc Spain

Candidate for the Degree of

Master of Science

Thesis: A STUDY ON LOG EVENT NOISE REDUCTION BY USING NAÏVE BAYES SUPERVISED MACHINE LEARNING

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in December, 2019.

Completed the requirements for the Bachelor of Science in Microbiology at the University of Oklahoma, Norman, Oklahoma in 1982.

Completed the requirements for the Bachelor of Science in Botany at the University of Oklahoma, Norman, Oklahoma in 1982.

Experience:

2015 – 2019, Senior Security Engineer/Project Manager III: Farmers/Zurich Insurance.
2013 – 2015, Server Specialist/Analyst/Security Engineer: State of Oklahoma, Department of Oklahoma Management of Enterprise Services.
2003 – 2013, Information Services Manager: State of Oklahoma, Oklahoma Real Estate Commission.
2001 – 2009, Consultant: Rattan Consulting.
2000 – 2001, Director of Infrastructure, MyWebCast.
1996 – 2000, Manager of Information Technology, Autocraft Industries.
1993 – 1996, Consultant, Rueb Group.